CompFUSE: Computational Framework for Unbiased Studies of Correlated Electron Systems

Ed D'Azevedo
Thomas Maier
Oak Ridge National Laboratory

FastMath Meeting

June 10, 2019

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

This work is supported by the Scientific Discovery through Advanced Computing (SciDAC) program funded by the US Department of Energy, Office of Science, Advanced Scientific Computing Research and Basic Energy Sciences, Division of Materials Sciences and Engineering.

CompFUSE Team

SciDAC
Scientific Discovery through Advanced Computing

Physics


Gonzalo Alvarez (ORNL)


Tom Berlijn (ORNL)


Steven Johnston (UTK)


Thomas Maier (ORNL)


Satoshi Okamoto (ORNL)


Douglas Scalapino (UCSB)

Math


Ed D'Azevedo (ORNL)


Feng Bao (FSU)


Clayton Webster (ORNL)

Computer Science


Wael Elwasif (ORNL)


Oscar Hernandez (ORNL)


Ying Wai Li (LANL)

Postdocs, students, staff


Arghya Chatterjee (ORNL)


Philip Dee (UTK/ORNL)


Peter Doak (ORNL)


Seher Karakuzu (ORNL/UTK)


Pontus Laurell (ORNL)


Peizhi Mai (ORNL)


Xuping Xie (ORNL)


Matthew Bachstein (UTK)


Aaron Kirby (UTK)

OAK RIDGE National Laboratory | 75 YEARS

THE UNIVERSITY OF TENNESSEE KNOXVILLE

UC Santa Barbara

FLORIDA STATE UNIVERSITY

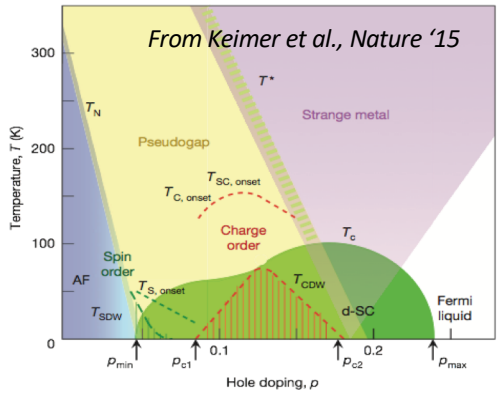
CompFUSE.ornl.gov

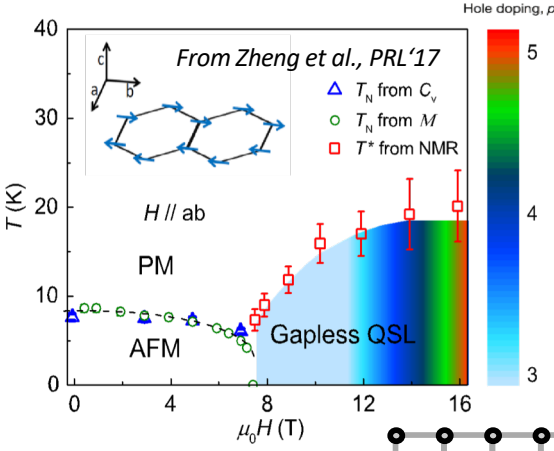
Strongly Correlated Electron Materials

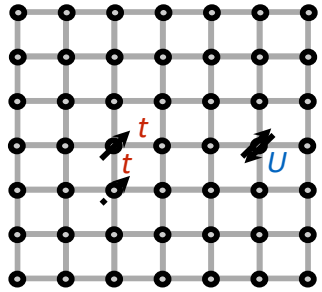
Overarching goal

Understand, predict and ultimately control the effects of correlations in quantum materials by developing a computational framework for controlled and unbiased studies of strongly interacting electron systems comprised of a diverse suite of complementary quantum many-body techniques

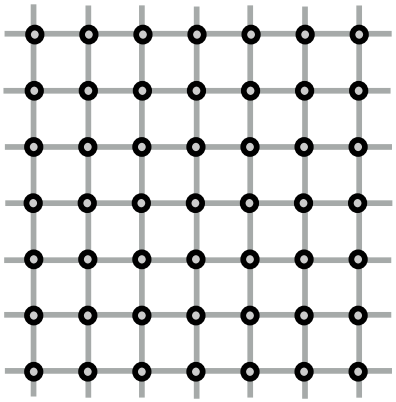
- **Unconventional (high-T_c) superconductors**
  - Cuprates, iron-based materials, …
  - Magnetism, superconductivity, charge order, …
  - Pairing mechanism?
- **Quantum spin liquids**
  - Honeycomb iridium oxides, ruthenium based materials, Herbertsmithite, …
  - Geometrically frustrated magnetic interactions
  - Stability of spin liquid ground state, Majorana fermions, …?
- **Model description**
  - Accuracy needed to describe effects of correlations and phase competition requires use of model Hamiltonians and complementary many-body methods, in conjunction with high-performance computing







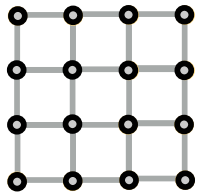
Numerical methods



Determinant Quantum Monte Carlo (DQMC)

- Finite size cluster
- Monte Carlo sampling of $Z = \mathrm{Tr} e^{-\beta \mathcal{H}}$
- Limited by Fermion sign problem

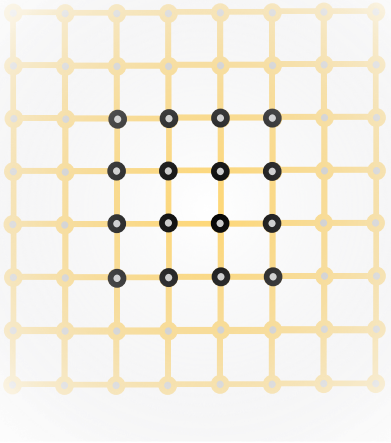
Blankenbecler et al., PRD '81.



Dynamic Cluster Approximation DCA(QMC)

- Cluster embedded in self-consistent host
- Monte Carlo sampling of $Z = \int \mathcal{D}[\phi^* \phi] e^{-S[\phi^*, \phi]}$
- Limited by Fermion sign problem (milder)

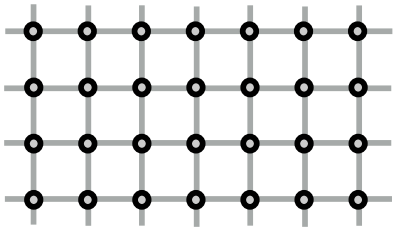
Maier et al., RMP '05.



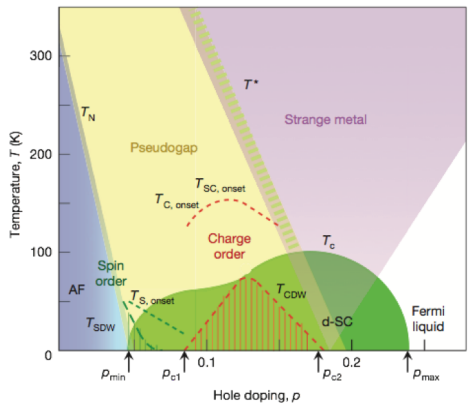
Density Matrix Renormalization group (DMRG)

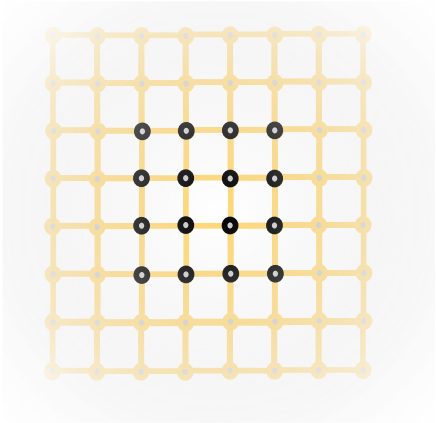
- Finite size, quasi-1D system
- Truncates Hilbert space based on density matrix
- Limited by entanglement entropy

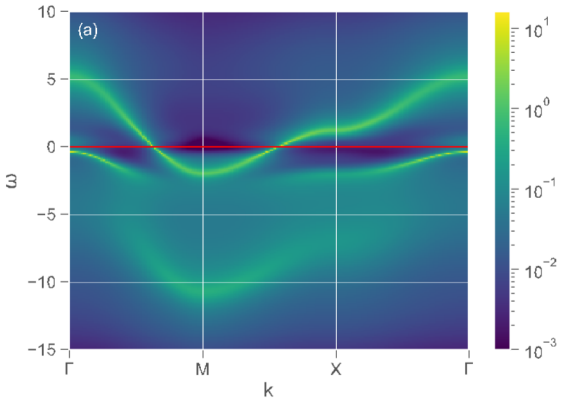
White et al., PRL '92.



CompFUSE: 3 Specific Aims







SA1: Advanced simulations of correlated quantum materials, including unconventional superconductors and quantum spin liquids.

SA2: Development of accelerated algorithms and efficient implementations, based on DQMC, DMRG and DCA.

SA3: Development of tools for verification and validation of simulations, in particular for analytic continuation of QMC data.

Focus: Dynamics, excited states, finite temperatures, mechanisms, 4-point scattering vertex Γ

Three aims facilitate ***controlled and reliable studies of correlated quantum materials*** using current ***petascale and future exascale computing*** architectures.

Computational Kernel of DQMC in DQMC Sweep

1) For $l = 1, 2, \cdots, L$
   a) For $i = 1, 2, \cdots, N$
      i) Flip $h'_{l,i} = -h_{l,i}$
      ii) Compute the Metropolis ratio

$$r_{l,i} = \frac{|M_+(h')|\,|M_-(h')|}{|M_+(h)|\,|M_-(h)|}$$

      iii) Acceptance condition (random number r and $r \sim \text{uniform}[0,1]$)

$$h_{l,i} \leftarrow h'_{l,i} \quad \text{if } r_{l,i} \geq r$$

2) Compute physical measurements

$$M_\sigma(h) = I + B_{L,\sigma}(h_L)B_{L-1,\sigma}(h_{L-1})\ldots B_{1,\sigma}(h_1)$$

$$B_{l,\sigma}(h_l) = e^{\sigma\nu\,\text{diag}(h_{l,1},h_{l,2},\ldots,h_{l,N})}\,e^{-\Delta\tau K}$$

Acceptance ratio related to computing determinant

If proposed move is accepted, the matrix is modified by a low rank update

Computational Challenges in Determinant Monte Carlo (DQMC)

- Many concurrent Monte Carlo "walkers" to fill device memory
- Need evaluation of determinants, det(M+uv') = (1 + v'inv(M)u) det(M)
- G = inv(M) = inv(I + B_L … B_2 B_1), where $B_i = V_i * B$, V_i is diagonal matrix but B is unchanged
- L * dt = 1/(k_B * T) is the inverse temperature, dt is the time step
- Size of matrix B is related to lattice size, O(100)
- Green's function matrix is modified by rank-1 update when a proposed event is accepted (Sherman-Morrison-Woodbury formula)
- Actual updates are delayed to perform rank-k update as matrix-matrix multiplication
- If number of time steps L is large (O(100)), then computing the matrix product B_L … B_2 B_1 is *extremely ill-conditioned*

Method using QR with column pivoting

1) Compute the pivoted QR: $B_1 = Q_1 R_1 P_1^T$
2) Set $D_1 = \mathrm{diag}(R_1)$ and $T_1 = D_1^{-1} R_1 P_1^T$
3) For $i = 2, 3, \cdots, L$
   a) Compute $C_i = (B_i Q_{i-1}) D_{i-1}$
   b) Compute the pivoted QR: $C_i = Q_i R_i P_i^T$
   c) Set $D_i = \mathrm{diag}(R_i)$ and $T_i = (D_i^{-1} R_i)(P_i^T T_{i-1})$
4) Compute $G = (T_L^{-T} Q_L^T D_b + D_s)^{-T} D_b Q_L^T$

At the last step, D_b and D_s are computed from D_L as

$$D_b(i) = \begin{cases} 1/|D_L(i)| & \text{if } |D_L(i)| > 1 \\ 1 & \text{otherwise} \end{cases}$$

$$D_s(i) = \begin{cases} D_L(i) & \text{if } |D_L(i)| \leq 1 \\ \mathrm{sgn}(D_L(i)) & \text{otherwise.} \end{cases}$$

Advancing Large Scale Many-body QMC Simulations on GPU Accelerated Multicore Systems, by A. Tomas, C-C Chang, R. Scalettar, Z. Bai, 2012 IEEE 26 IPDP

Variant to reduce the cost of QR with pivoting (DGEQP3) by forming cluster of matrices

$B_L \ldots B_2 B_1 = F_l \ldots F_2 F_1$

$F_1 = B_s \ldots B_2 B_1$
$F_2 = B_{2s} \ldots B_{s+2} B_{s+1}$

Pre-pivoting method to use QR

1) Compute the pivoted QR: $B_1 = Q_1 R_1 P_1^T$
2) Set $D_1 = \mathrm{diag}(R_1)$ and $T_1 = D_1^{-1} R_1 P_1^T$
3) For $i = 2, 3, \cdots, L$
   a) Compute $C_i = (B_i Q_{i-1}) D_{i-1}$
   b) Compute the permutation P_i such as $\widehat{C}_i = C_i P_i$ has decreasing norm columns
   c) Compute a regular QR: $\widehat{C}_i = Q_i R_i$
   d) Set $D_i = \mathrm{diag}(R_i)$ and $T_i = (D_i^{-1} R_i)(P_i^T T_{i-1})$
4) Compute $G = (T_L^{-T} Q_L^T D_b + D_s)^{-T} D_b Q_L^T$

Pre-pivoting method gives small relative differences to QRP on small problems

Accurate Solution via SVD

$U_1 D_1 V_1' = B_1$ (compute SVD)

For j=2,3, …, L do

$C_j = (B_j U_{j-1}) D_{j-1}$

$U_j D_j V_j = C_j$ (compute SVD)

Enddo

Decompose $D_L = D_b Ds$,

Where D_b = entries > 1 or 1, D_s = entries < 1 or 1

$M = [I + U_L D_L (V_1 V_2 \ldots V_L)')$

$M = [I + U_L D_b D_s V')$, $V = V_1 V_2 \ldots V_L$

$M = inv(D_b) U_L [inv(D_b) U'_L + D_s V']$

$G = inv(M) = inv(inv(D_b) U'_L + D_s V') U_L' D_b$

Stable solutions of linear systems involving long chain of matrix multiplications, Z Bai, C Lee, R-C Li, S Xu,
Linear Algebra and its Applications, 435(3), 2011, p659-673

Computational Needs

Batch DGEQRP (QR with pivoting) on GPU

Batch SVD on GPU

Batch communication avoiding DGEQRP on GPU ??

Batch DGEQRF already available in CUBLAS and MAGMA

Also need batch routines to apply "Q" on GPU (DORGQR)

Computational Challenges in DCA++

DCA++ Monte Carlo code is collaboration with ETH and ORNL.

DCA++ has been finalist for Gordon Bell and optimized for Titan and Summit and able to use mixed FP32/FP64 precision

Self-consistency loop with 2 primary kernels:

- (1) Coarse graining of single particle Green's function to reduce complexity of infinite size lattice to an effective finite cluster problem
- (2) Quantum Monte-Carlo based solution of effective cluster problem

Kernel (1) requires global reductions and dense matrix operations to compute matrix inverses

Kernel (2) has computational needs in:

- delayed submatrix-updates
- mapping of continuous time measures to Fourier representation, which requires non-uniform FFT or direct (batch) multiplication by Fourier matrix

Details

Green's matrix is diagonal in Fourier space $G_{\ell_1,\ell_2} = G^0_{\ell_1}\delta_{\ell_1\ell_2} - G^0_{\ell_1} M_{\ell_1,\ell_2} G^0_{\ell_2}$

Matrix M is related to measurements at different real-space and random times

Transform M to Fourier space:

- Direct matrix multiplication by Fourier matrix for small case
- Interpolate M (convolution with Gaussian-like function) onto regular grid

$$\mathcal{M}_{(l_1 r_1 \tau_1),(l_2 r_2 \tau_2)} = \sum_{\tau_1',\tau_2'} \phi_{\tau_1,\tau_1'} M_{(l_1 r_1 \tau_1'),(l_2 r_2 \tau_2')} \phi_{\tau_2',\tau_2}$$

- Perform 2D FFT

$$\mathcal{M}_{(l_1 r_1 \omega_1),(l_2 r_2 \omega_2)} = \mathrm{FFT}_{2\mathrm{D}}[\mathcal{M}_{(l_1 r_1 \tau_1),(l_2 r_2 \tau_2)}]$$

- Recover Fourier coefficients of M

$$M_{(l_1 r_1 \omega_1),(l_2 r_2 \omega_2)} = \frac{1}{\phi_{\omega_1}} \mathcal{M}_{(l_1 r_1 \omega_1),(l_2 r_2 \omega_2)} \frac{1}{\phi_{\omega_2}}$$

Future Needs

Computation of 4-point Green's function G4 $G^4_{\ell_1,\ell_2,\ell_3,\ell_4} = \langle G_{\ell_1,\ell_4} G_{\ell_2,\ell_3} - G_{\ell_1,\ell_3} G_{\ell_2,\ell_4} \rangle_{MC}$

Translation and lattice symmetrice reduce to 3-index array, q = $k_1 + k_2 = k_3 + k_4$
G4(k_1,k_2,k_3,k_4) = G4(k+q,-k,-k',k'+q) or as G(q,k,k')

Further calculations with G4 for dual self energy (maybe rich in convolutions)

$$\delta\widetilde{\Sigma}^{(2)}_{K_1,K_2}(\tilde{\mathbf{k}}) = -\frac{1}{N_c N\beta}\sum_{Q,\tilde{\mathbf{q}}} \gamma_{Q,K_1,K_2}\widetilde{G}^{(0)}_{Q-K_2,Q-K_1}(\tilde{\mathbf{q}}-\tilde{\mathbf{k}}) \quad - \frac{1}{N_c^2 N^2\beta^2}\sum_{Q_1,K_3} \gamma_{Q_1,K_1,K_3}\ F(Q_1,K_1,K_3,K_2,\tilde{\mathbf{k}})$$

$$\gamma_{Q,K_1,K_2} = \beta N_c \frac{G^4_{Q,K_1,K_2} + G^c_{K_1} G^c_{Q-K_1}(\delta_{K_1,K_2} - \delta_{K_1,Q-K_1})}{G^c_{K_1} G^c_{K_2} G^c_{Q-K_1} G^c_{Q-K_2}}$$

$$F(Q_1,K_1,K_3,K_2,\tilde{\mathbf{k}}) = \sum_{Q_2,K_4} C(Q_1,K_1,K_3,Q_2,K_4,K_2,\tilde{\mathbf{k}})\ \gamma_{Q_2,K_4,K_2}$$

$$C(Q_1,K_1,K_3,Q_2,K_4,K_2,\tilde{\mathbf{k}}) = \sum_{\tilde{\mathbf{q}}} E(Q_1,K_3,Q_2,K_4,\tilde{\mathbf{q}})\widetilde{G}^{(0)}_{Q_1-K_1,Q_2-K_2}(\tilde{\mathbf{q}}-\tilde{\mathbf{k}})$$

$$E(Q_1,K_3,Q_2,K_4,\tilde{\mathbf{q}}) = \sum_{\tilde{\mathbf{k}'}} \widetilde{G}^{(0)}_{K_3,K_4}(\tilde{\mathbf{k}'})\widetilde{G}^{(0)}_{Q_1-K_3,Q_2-K_4}(\tilde{\mathbf{q}}-\tilde{\mathbf{k}'})$$

Computational Needs in DCA++

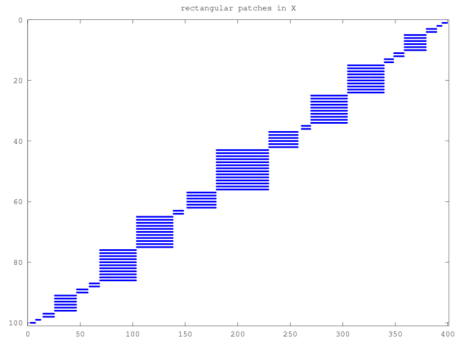
Batch Non-uniform FFT on GPU

Batch 2D FFT in FP32 that can take advantage of tensor cores (FP16)

- Radix-4 Fourier 4x4 matrix consists of {-1,0,1} (related to cos(90), sin(90) exactly presentable in FP16
- Radix-2 Fourier 2x2 matrix is real and consist of {1,-1}, exactly presentable in FP16
- Radix-8 Fourier 8x8 matrix has sqrt(2)/2 = cos(45) ~ 985/1393 approximated as rational integer representation in FP16
- Split X_fp32(:) = X1_fp16(:) + (2^(-10)) * X2_fp16(:), then
  FFT(X_fp32) = FFT(X1_fp16) + (2^(-10)) * FFT(X2_fp16)

Library and algorithms for distributed array to hold 3-index G4, gamma and convolution calculations for dual self energy

Computational Challenges in Density Matrix Renormalization Group (DMRG++)



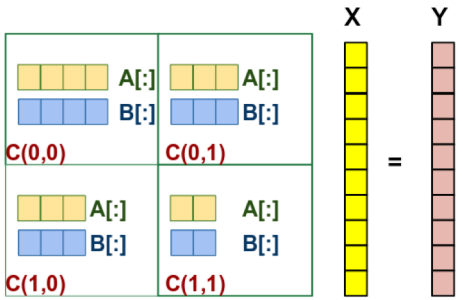
Goal to find few lowest eigen pairs of Hamiltonian by finding a suitable subspace. Iteration by considering "left" and "right" parts of lattice.

Matrix-vector multiplication in Lanczos method is dominant kernel

Hamiltonian is expressed as sum of Kronecker of operators

$$H' = H_L \otimes I_R + I_L \otimes H_R + \sum_{\gamma=0}^{\gamma<\Gamma} c_L^{\gamma} \otimes c_R^{\gamma}$$

Hamiltonian has symmetry property and conservers quantum numbers



Key insight: group states by quantum numbers (as patches) to express Hamiltonian as many sums of Kronecker products of small dense matrices

Several nested levels of parallelism, challenges with load balancing

Kronecker Product

If matrix A is m by n, matrix B is p by q, C = kron(A,B) is (m*p) by (n*q)

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}$$

$$(\mathbf{A} \otimes \mathbf{B})^{\mathsf{T}} = \mathbf{A}^{\mathsf{T}} \otimes \mathbf{B}^{\mathsf{T}}$$

$$(\mathbf{A} \otimes \mathbf{B})^{*} = \mathbf{A}^{*} \otimes \mathbf{B}^{*}.$$

Implementation as Batched DGEMMs

Recall vec(Y) = kron(A,B) * vec(X) can be evaluated as Y = B * X * transpose(A)

Hamiltonian is block partitioned, $C[I,J] = \sum_k A_{IJ}^{(k)} \otimes B_{IJ}^{(k)}$

Let $W_{IJ}^{(k)} = B_{IJ}^{(k)} * X[J]$

$$C[I,J] * X[J] = \left(\sum_k^K A_{IJ}^{(k)} \otimes B_{IJ}^{(k)} \right) * X[J]$$

$$= \sum_k^K (B_{IJ}^{(k)} * X[J] * (A_{IJ}^{(k)})^t)$$

$$= \sum_k^K (W_{IJ}^{(k)} * (A_{IJ}^{(k)})^t)$$

$$[W_{IJ}^1 | W_{IJ}^2 | \dots | W_{IJ}^K] = [B_{IJ}^1 | B_{IJ}^2 | \dots | B_{IJ}^K] * X[J]$$

$$\mathcal{W}_{IJ} = \mathcal{B}_{IJ} * X[J] \ .$$

$$Z[I,J] = C[I,J] * X[J]$$

$$= [W_{IJ}^1 | W_{IJ}^2 | \dots | W_{IJ}^K] * [A_{IJ}^1 | A_{IJ}^2 | \dots | A_{IJ}^K]^t$$

$$= \mathcal{W}_{IJ} * (\mathcal{A}_{IJ})^t$$

$$Y[I] = \sum_J Z[I,J]$$

$$= \sum_J (\mathcal{W}_{IJ} * \mathcal{A}_{IJ}^t)$$

$$= [\mathcal{W}_{I1} | \dots | \mathcal{W}_{I,N_p}] * [\mathcal{A}_{I1} | \dots | \mathcal{A}_{I,N_p}]^t$$

Computational Needs of DMRG++

Hamiltonian is Hermitian, C[I,J] = conj(transpose(C[J,I])) = C[J,I]'

Recall conj(transpose(kron(A,B))) = kron(conj(transpose(A)), conj(transpose(B)))

Vec(Y)) = kron(A',B') * vec(X), evaluated as Y = (B') * X * transpose(A') = (B')*X*conj(A)

Would be nice to take advantage of symmetry by storing only triangular part in GPU device memory.

Need pure conjugate option (not conjugate transpose) in ZGEMM (possible work around by storing transpose of A in memory).

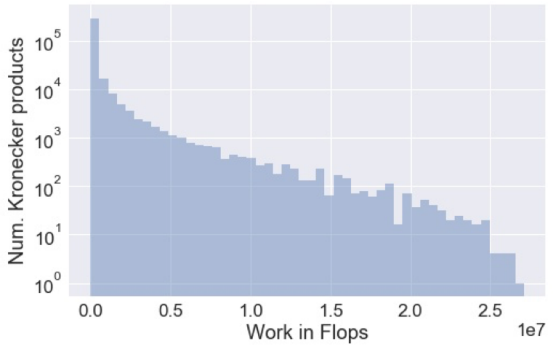
Need option for atomic update in batched GEMM for C += a*A*B

- Volta and Pascal GPU has hardware atomic update for FP64 and FP32
- Atomic add needed only when storing "C(I,j)" from shared memory to global memory and only when "beta=1" for C += a*A*B
- Only very minor changes in code for MAGMA batched BLAS
- May also be useful for element-by-element formulation in FEM for matrix-vector multiply

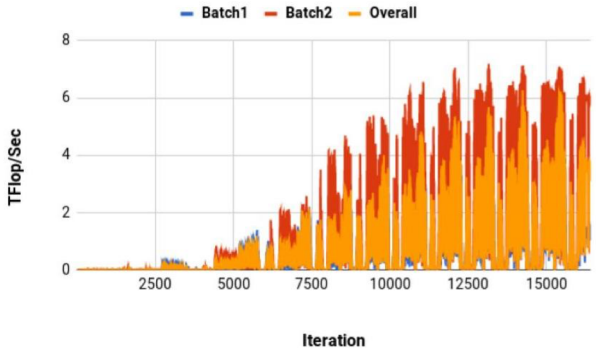
backup

Efficient Implementations

Nested parallelism: block rows, block columns, sums of Kronecker products, matrix-matrix operations
Significant imbalances in workload
Task-based OpenMP parallelism
Batched DGEMM operations on GPU using MAGMA library
15X speedup (over start of project) on multicore CPUs
6 Tflops (FP32) on GPU



Histogram of work intensity in Hamiltonian matrix



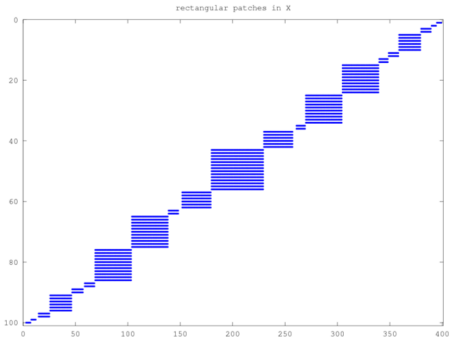
Batched GEMM performance (FP32) in DMRG++ on Volta GPU

Reformulation as Kronecker Products

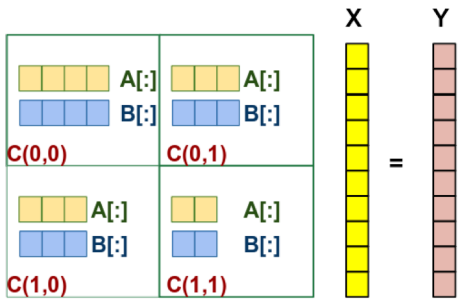
Hamiltonian matrix in DMRG is expressed as Kronecker product of operators

$$H' = H_L \otimes I_R + I_L \otimes H_R + \sum_{\gamma=0}^{\gamma<\Gamma} c_L^{\gamma} \otimes c_R^{\gamma}$$

Hamiltonian has symmetry property and conserves quantum numbers
Key insight: group states by quantum numbers (as patches) to express Hamiltonian as many sums of Kronecker products of small dense matrices



Acceptable states (organized by quantum number) form non-overlapping patches.



Block partitioning of Hamiltonian matrix.
Each submatrix is sum of Kronecker products.